PRICE $ 2.00

THE 80 NOTEBOOK

MAY 1980 ISSUE #2

A PROGRAM ABOUT THE PLANETS

In the past few years, many people have learned of a most unique event upcoming in our solar system. In 1982, all the planets except Pluto, the outermost planet, will line up in a straight line! This event is commonly called " The Grand Alignment".

According to some theories, this event may have formed the "Star of Bethlehem" during the birth of Jesus Christ because of the combined effects of seeing all the visible planets in the night sky together in a tight cluster just before alignment and because this event only occurs once every 2000 years!

These modern theories suggest that this alignment of the planets will cause many disasterous changes on Earth because of the gravitational effects of such an alignment. When all the planets in the solar system align with each other, the gravitational pull of each planet in the chain is additive and can effect the orbits of the planets.

These gravitational effects can cause volcanic activity, earthquakes, tidal waves, climate changes, changes in the moon's orbit and possibly change the length of an Earth day!

The dramatic increase in the number of major earthquakes, floods, volcanic activity and severe winters in the past 20 years would seem to bear out some of these possibilities as we approach the alignment!

In the following program, we will look at some of the various statistics concerning the planets in our solar system and perform a simple simulation of the grand alignment and similar events.

In lines 5 thru 140, the system establishes the various statistics known about the sun and each of the 9 planets in our solar system. These statistics include the name of the planet, its distance from the sun in kilometers, the diameter of the planet in kilometers, the length of the planet's year in Earth days, the number of moons orbiting the planet, the length of the planet's day in Earth hours, the mass of the planet in grams, the surface temperature of the planet in degrees centigrade, the density of the planet in grams per cubic centimeter, the surface gravity of the planet relative to Earth's gravity, the volume of the planet in cubic centimeters, and the escape velocity from the planet in kilometers per second.

This section of the program also asks for the number of sectors in an orbit. This value is used to divide a planet's orbit, expressed in Earth days revolution time, into sectors so that the program can determine where a planet is relative to the other planets.

This sector location changes relative to time so the program assumes the base year to be 1982 with all the planets in the same sector of space. The number of sectors in an orbit may range from 4 to 360 depending on the degree of accuracy desired in determining planetary alignment.

Whenever the program displays the statistics on a planet, it also computes the number of planets in alignment, the sector where the planet currently resides and the year the computer is sitting at relative to its calculations.

Functionally, the program allows 2 options. The first option allows you to enter a year and have the computer determine the position of the planets in that year. Initially, the program asks you which planet's statistics you are interested in followed by the year you are interested in. Once the computations are made, the planetary statistics are printed along with sector and alignment information based on the last number of sectors per orbit value inputted.

The second option allows the user to determine in what year the next alignment of a group of consecutive planets will take place starting from the last year the computer displayed planetary statistics for.

You must tell the computer the first planet, the one closest to the sun, and the last planet, the one farthest from the sun, that form the range of consecutive planets that must be in alignment in a particular year.

You must also input the desired number of sectors per orbit value to be used to determine the tolerance in the alignment calculations.

Additionally, you must tell the computer which direction in time you wish the program to go from the current year in looking for the desired alignment, either into the future or into the past.

Once the desired alignment is found, the program will display the planetary statistics, year and alignment data for the first planet in the alignment planetary chain.

If, during the execution of either the first or second option, you wish to cancel the option in progress, just hit any key on the keyboard and the computer will stop and display the planetary statistics for the year it was last working on in either option's required calculations since both option's calculations advance through solar time at the rate of one Earth year per cycle.

The Solar System program is available on cassette for $3.49 to those who do not wish to key the program into their TRS-80 by hand. The program requires 4K and is written in Level 2 BASIC.

```
5 CLEAR3000
10 REM THE SOLAR SYSTEM
20 REM COPYRIGHT 1980, THE 80 NOTEBOOK
```

```
 30 DIMA$(9),B(9,11):DEFINTI,O,N,J
 40 A$(0)="SUN":B(0,1)=0:B(0,2)=1392000:B(0,3)=0:B(0,4)=9:B(0,5)=609.12:B(0,6)=33
2946:B(0,7)=5600:B(0,8)=1.469:B(0,9)=27.9:B(0,10)=1303600:B(0,11)=617.5
 50 A$(1)="MERCURY":B(1,1)=57910000:B(1,2)=4879:B(1,3)=87.969:B(1,4)=0:B(1,5)=140
8.8:B(1,6)=.056:B(1,7)=400:B(1,8)=5.5:B(1,9)=.381:B(1,10)=.056:B(1,11)=4.27
 60 FORI=0TO9:B(I,0)=I:NEXTI:?=1582:N=3
 70 A$(2)="VENUS":B(2,1)=108210000:B(2,2)=12100:B(2,3)=224.701:B(2,4)=0:B(2,5)=58
32:B(2,6)=.815:B(2,7)=475:B(2,8)=5.25:B(2,9)=.9032:B(2,10)=.8572:B(2,11)=10.36
 80 A$(3)="EARTH":B(3,1)=149600000:B(3,2)=12756:B(3,3)=365.256:B(3,4)=1:B(3,5)=23
.93:B(3,6)=1:B(3,7)=30:B(3,8)=5.517:B(3,9)=1:B(3,10)=1:B(3,11)=11.18
 90 A$(4)="MARS":B(4,1)=227940000:B(4,2)=6790:B(4,3)=686.98:B(4,4)=2:B(4,5)=24.62
:B(4,6)=.1074:B(4,7)=-10:B(4,8)=3.94:B(4,9)=.3799:B(4,10)=.1504:B(4,11)=5.03
100 A$(5)="JUPITER":B(5,1)=778340000:B(5,2)=142800:B(5,3)=4332.59:B(5,4)=12:B(5,
5)=9.84:B(5,6)=317.89:B(5,7)=-130:B(5,8)=1.33:B(5,9)=2.643:B(5,10)=1318.7:B(5,11
)=60.22
110 A$(6)="SATURN":B(6,1)=1427010000:B(6,2)=119300:B(6,3)=10759.2:B(6,4)=10:B(6,
5)=10.23:B(6,6)=95.14:B(6,7)=-150:B(6,8)=.706:B(6,9)=1.159:B(6,10)=743.6:B(6,11)
=36.25
115 INPUT"NUMBER OF SECTORS/ORBIT";O:IFO(40ORO)360THEN115
120 A$(7)="URANUS":B(7,1)=2869600000:B(7,2)=47100:B(7,3)=30685:B(7,4)=5:B(7,5)=1
0.82:B(7,6)=14.52:B(7,7)=-190:B(7,8)=1.7:B(7,9)=1.11:B(7,10)=47.1:B(7,11)=22.4
130 A$(8)="NEPTUNE":B(8,1)=4496700000:B(8,2)=49400:B(8,3)=60190.2:B(8,4)=2:B(8,5
)=15.8:B(8,6)=17.25:B(8,7)=-220:B(8,8)=1.77:B(8,9)=1.21:B(8,10)=53.7:B(8,11)=23.
9
140 A$(9)="PLUTO":B(9,1)=5898900000:B(9,2)=5900:B(9,3)=90465.2:B(9,4)=0:B(9,5)=1
53.36:B(9,6)=.1:B(9,7)=-240:B(9,8)=5.5:B(9,9)=.47:B(9,10)=.1:B(9,11)=5.1
150 F$="":P=INT((B(J,0)/B(J,3))*40):J=0
160 FORI=2TO9
170 R=INT((B(I,0)/B(I,3))*40)
180 IFR<>PTHEN210
190 IFJ=0THENF$=A$(I-1)+" ":J=1
200 F$=F$+A$(I)+" ":GOTO220
210 P=R:IFJ=1THEN230
220 NEXTI
230 CLS
240 PRINT"PLANETS ALIGNED - ";F$
250 PRINT"SELECTED PLANET - ";A$(M)
260 PRINT"DISTANCE FROM THE SUN IN KILOMETERS ";B(N,1)
270 PRINT"DIAMETER IN KILOMETERS ";B(N,2)
280 PRINT"LENGTH OF YEAR IN EARTH DAYS ";B(N,3)
290 PRINT"NUMBER OF MOONS ";B(N,4)
300 PRINT"LENGTH OF DAY IN EARTH HOURS ";B(N,5)
310 PRINT"MASS IN GRAMS ";B(N,6)*5.977E+27
320 PRINT"SURFACE TEMPERATURE IN CENTIGRADE ";B(N,7)
330 PRINT"DENSITY IN GRAMS/CUBIC CENTIMETERS ";B(N,8)
340 PRINT"SURFACE GRAVITY RELATIVE TO  EARTH ";B(N,9)
350 PRINT"VOLUME IN CUBIC CENTIMETERS ";B(N,10)*1.08321E+27
360 PRINT"ESCAPE VELOCITY IN KILOMETERS/SECOND ";B(N,11)
370 PRINT"YEAR ";Z;"SECTOR ";INT((B(N,0)/B(N,3))*40);" (HIT ENTER)"
380 F$="":F$=INKEY$:IFF$="" "ORF$=""THEN350
400 CLS:PRINT"1) ADVANCE TO NEW YEAR"
410 PRINT"2) LOOK FOR ALIGNMENT":PRINT"3) END"
420 INPUT"ENTTER CHOICE ";J:IFJ<1ORJ>3THEN420
425 IFJ=3THENEND
430 IFJ=2THEN1000
435 CLS
440 FORI=0TO9:PRINTI;")";A$(I):NEXTI
450 INPUT"ENTER CHOICE";N:IFN<0ORN>9THEN450
460 INPUT"ENTER YEAR";K:IFK=2THEN150
470 FORI=1TO9
473 IFK<2THEN483
475 B(I,0)=B(I,0)+B(I,3)
480 IFB(I,0)>B(I,3)THENB(I,0)=B(I,0)-B(I,3):GOTO480ELSE495
483 B(I,0)=B(I,0)-B(I,3)
485 IFB(I,0)<1THENB(I,0)=B(I,0)+B(I,3):GOTO485
495 NEXTI
500 IFK<2THENZ=Z-1ELSEZ=Z+1
505 F$="":F$=INKEY$:IFF$="" "ORF$=""THEN510ELSE150
510 IFK<2THEN470ELSE150
1000 CLS:FORI=1TO9:PRINTI;")";A$(I):NEXTI
1010 INPUT"FIRST PLANET IN ALIGNMENT ";J:IFJ<1ORJ>8THEN1010
1020 INPUT"LAST PLANET IN ALIGNMENT ";K:IFK<J+1ORK>9THEN1020
1025 INPUT"NUMBER OF SECTORS/ORBIT";O:IFO<40ORO>360THEN1025
1026 N=J
1030 CLS:PRINT"1) INTO THE FUTURE":K=INT(K)
1040 PRINT"2) INTO THE PAST"
1050 INPUT"ENTER CHOICE ";R:IFR<1ORR>2THEN1050
1060 FORI=1TO9
1065 IFR=2THEN2000
1070 B(I,0)=B(I,0)+B(I,3)
1080 IFB(I,0)>B(I,3)THENB(I,0)=B(I,0)-B(I,3):GOTO1080ELSE2000
2000 B(I,0)=B(I,0)-B(I,3)
2010 IFB(I,0)<1THENB(I,0)=B(I,0)+B(I,3):GOTO2010
3000 NEXTI:P=INT((B(J,0)/B(J,3))*40)
3010 IFR=2THENZ=Z-1ELSEZ=Z+1
3015 F$="":F$=INKEY$:IFF$="" "ORF$=""THEN3020ELSE150
3020 FORI=JTOK:J=INT((B(I,0)/B(I,3))*40)
3030 IFJ<>PTHEN1060
3040 NEXTI:GOTO150
```

NEW PRODUCTS

DISK KEYPLUS: AN INTEGRATED UTILITY PACKAGE FOR THE TRS-80

Disk Keyplus is a powerful collection of utilities that can be enabled directly from the keyboard of the TRS-80. Carefully designed to maximize ease of use, all Disk Keyplus routines may be turned on or off in just two key strokes.

Disk Keyplus supports auto-repeat, lowercase video (optional hardware modification required), restoration of lost BASIC programs, single key stroke user definable strings, BASIC shorthand, direct graphic character input, typewriter style input, and more!

Disk based utilities include a routine that generates a previously defined string three different ways: at power up, during Keyplus initiation, or at the stroke of just two keys. More flexible than the DOS AUTO command, Disk Keyplus will execute any combination of commands and/or programs.

Another routine allows users to initialize Disk Keyplus with any combination of utilities enabled or disabled.

Disk Keyplus may be used with either TRSDOS or NEWDOS. A cassette with both the 32K and 48K versions is available for only $19.95.

Non-disk Keyplus (Lv.2 16K) is available for only $14.95. PA residents add 6% sales tax.

S J W, Inc. P.O. Box 438 Huntingdon Valley, PA 19006 (215) 947-2057

TRS-80 MODEL II EDITOR ASSEMBLER

EDAS 4.0 has just been released by Galactic Software ltd. EDAS is a RAM-resident text editor and assembler for the TRS-80 Model II running under TRSDOS. The editor provides text editing facilities for the modification of alphanumeric text files. Command syntax is identical to the Model II's Disk BASIC editor. EDAS also provides text block move, global change, string search, and line scroll capabilities.

The assembler portion of EDAS facilitates the translation of Z-80 symbolic language (ZILOG mnemonics) source code programs into machine executable code. Assembler switches provide the user with options to suppress source and symbol table listings, suppress object code generation, output the assembled code directly to memory or disk, and more.

All TRSDOS commands are directly executable from within EDAS. This feature gives you the capability of displaying directories, listing files, setting FORMS, or any other command without exiting the environment of EDAS. Interfacing to DEBUG has been provided to enable a direct approach to debugging user generated code.

Great amounts of time and effort were expended to give the user of this Editor Assembler the absolute best in ease of operation and functional efficiency. Optimize assembly programming time, with the Editor Assembler designed with the programmer in mind. EDAS is available NOW for $229.00.

Galactic Software ltd., 11520 N. Port Washington Rd., Mequon, WI 53092 (414) 241-8030

.....

MICROCOMPUTERS SEEN AS BOON TO CIVIL ENGINEERS AND SURVEYORS

The use of microcomputers to perform the calculations needed to prepare subdivision maps and improvement plans for the building industry is having an extremely beneficial effect on the industry as a whole, according to Andrew Machen, a Registered Civil Engineer who has been involved in land development in Southern California for the past 18 years, and who has developed a subdivision geometry software package for microcomputers that he has been using in his engineering practice since 1977.

"Historically," Machen said, "engineers have used mechanical calculators, slide rules, logarithms and tables of trigonometric functions to perform the myriad of computations necessary to establish property boundaries and street alignments, the sizing of water supply systems and sewage collection facilities and the designing of storm drainage facilities. Electronic calculators having the necessary functions to perform the required calculations have become generally available only within the last ten years, and the high initial cost of large computer systems coupled with the necessity to hire specially trained operators and programmers has been a deterrant to the use of computers by many development designers.

Now, however, microcomputer technology has made it economically feasible for engineering firms of all sizes to provide computing facilities for its design professionals. Calculations that formerly required hours to perform manually, can now be performed in minutes by simply keying in a few input parameters, and letting the computer do the repetitious calculations. An added bonus to the small engineering or surveying office is the availability of general business management software including word processing, accounts receivable, client billing and cost estimating.

Machen's subdivision geometry software, along with an instruction manual, is available for a price of $260, and includes traverse geometry, profile grade calculations including straight grades and parabolic curves, intersections and triangle solutions, circular curve geometry and field staking, and has radial stakeout capabilities. For a free brochure, write to Andrew Machen, Consulting Civil Engineer, 143C S. Cedros, Solana Beach, CA 92075, or call (714) 755-4033.

.....

UNCLASSIFIED ADS:

I am stuck out in the sticks here in Mississippi and would like to purchase your used TRS-80 cassette programs. Jim Nichols, Box 249, University, MS 38677. (601) 234-2167

.....

NOTICE: Past issues of THE 80 NOTEBOOK are available for $2.00 per copy from THE 80 NOTEBOOK, R.D.#3 Box 192A, Nazareth, PA 18064.

PROGRAMMING - - BASIC, WHAT ARE THEY?!

A great many of the purchasers of TRS-80 equipment are new comers to the field of computers and programming and, as such, need a practical education in both of these areas in order to fully appreciate their machines and use them to their best advantage.

With this article, I will begin a series of tutorial articles designed to educate the novice as to how a computer functions, what a program is, what is BASIC, Level I BASIC instructions, Level II BASIC instructions, disk BASIC instructions, TRSDOS and how it functions, and how to write programs to perform specific tasks on your computer such as accounting functions, scientific problems and simulations.

Other future tutorial articles will examine, in detail, how computers can be used in school work, what is Assembly Language, and how to write an Assembly Language program that can be called from a BASIC program.

To start with, you should have a detailed understanding of what a computer is. A computer is composed of three main components: an arithmetic/logic unit, memory and a control unit. This should not be confused with the external appearance of your TRS-80.

The actual computer that the TRS-80 uses is housed inside the keyboard unit. It is the CRT screen and the keyboard itself that make up most of the bulk of your TRS-80. Both are needed in order for the computer to communicate with the outside world - you, its user!

Getting back to the computer's components, neither one of nor any pair of these three main components by them selves is useful. Only working together, can they accomplish a task.

First, let's look at memory. Memory is the component in which the computer stores information. This information can consist of numbers, text and instructions by which the computer can use its information.

By itself, the computer is powerless! It is up to the user to supply it with the instruction needed to perform a task and then, the numbers and/or text needed to carry out its instructions. In the TRS-80, these instructions and related information are given to the computer through the keyboard and/or the cassette tape player.

Memory is divided up into many individual storage units called bytes. Each byte can hold a single character. Therefore, several consecutive bytes in memory must be used to store large numbers, portions of text, or instructions to the computer.

Second, the arithmetic/logic unit provides the computer with the ability to understand and perform the instructions given to it. These instructions can only be in the machine's own language and are of a very simple nature.

Some instructions move information from one place in memory to another, others perform arithmetic, others input information from tape or the keyboard, others output information onto tape or the CRT screen, and some compare one area of memory to another in order to decide if something else is to be done.

The way a user communicates these machine language instructions to the computer is thru Assembly Language. Assembly Language is merely a symbolic representation of the arithmetic/logic unit's set of machine language instructions.

However, the TRS-80 uses another language - BASIC. The way the TRS-80 is able to understand the BASIC language is thru a special section of memory held within the keyboard of your TRS-80.

This special section of memory contains the series of machine language instructions needed to understand instructions in the BASIC language and to convert these BASIC instructions into an appropriate series of machine language instructions to accomplish the task implied by the BASIC instructions.

For example, when you type in CLOAD, the TRS-80 invokes a series of machine language instructions which will read a group of BASIC instructions from the cassette tape player into memory. While CLOAD is only a simple BASIC instruction, the resulting machine language instructions performed to accomplish that implied task number in the hundreds!

Lastly, the control unit is the component which governs or controls the use or action of the other two components and is responsible for transferring information to and from the outside world when requested by the arithmetic/logic unit.

For example, when the arithmetic/logic unit needs a piece of information from memory, the control unit fetches the information and restores it when the arithmetic/logic unit is finished with it. When input or output of information is requested, the control unit performs the transfer of information to or from memory.

Before a machine language instruction is performed, it is fetched from memory by the control unit which passes it to the proper section of the arithmetic/logic unit built to perform that specific instruction.

With this in mind, let's take a look at the Level I BASIC language and its capabilities:

First we should look at the way computer memory is used in BASIC. A portion of memory is used to store the series of BASIC statements which make up your BASIC program (a program being a series of instructions which, when carried out, perform a specific task). The remainder of available memory can be used to store information your program may need in order to carry out its task.

This information must be defined in a series of individually addressable pieces called fields (such as name, employee number, and rate in payroll information). These fields must be addressed in your program by unique names.

Numeric fields may use the letters of the alphabet as names (letters A thru Z; for example - "A" might contain hours, "B" might contain rate, and so on).

Alphabetic fields such as your name are called string fields. In a BASIC program, you can only have two string fields named A$ and B$. Each field may contain a piece of information up to 16 characters in length.

Numeric fields may have up to 6 significant digits each. Additionally, your BASIC program may use a table of related numeric fields (such as a table of rates sequenced in order by employee number) by referencing the table by the name A(N), where N is a numeric field containing the number of the table field you wish to address (relative to 1).

In general, the BASIC statements in your program are composed of a statement number followed by a BASIC instruction or command (these statement numbers sequence the order in which the statements are carried out in your program).

The difference between a BASIC command and an instruction is that a command stops the execution of a program containing it after it is carried out, while an instruction is carried out and the program continues with the next statement in the program.

Because of this, commands are not normally used in a program, but are used in the "READY" state of the computer. This is the condition that exists when the TRS-80 is not running a user program. At this time, BASIC commands which can LOAD, SAVE or LIST programs can be entered through the keyboard and carried out immediatly.

BASIC instructions without their usual numbers can also be entered through the keyboard at this time to be carried out immediatly. This type of instruction execution is rather limited because the instruction can not take advantage of related information and instructions normally possible in a BASIC program. Because of this, the "READY" state execution of BASIC instructions is most often used to print the answer to some equation entered, thus using the computer as a calculator.

Next, let's look at the various BASIC commands in Level 1 BASIC:

The "NEW" command clears the computer's memory in preparation for inputting a BASIC program, statement by statement, through the keyboard.

After a program has been inputted through the keyboard, you will want to copy your program onto a cassette tape so that it can be easily loaded and run when desired. To do this, you should use the "CSAVE" command. This process also requires the user to assign a single letter name for your program as stored on the cassette tape. To do this, the format of the CSAVE command as inputted through the keyboard must be CSAVE"A", where "A" is the letter by which the program will be identified on the cassette tape.

When you wish to use a program stored on cassette tape, you should use the "CLOAD" command. If the particular cassette you are working with has more that one program stored on it, you must specify the letter which identified your program during the CSAVE process. To do this, the format of the CLOAD command as inputted through the keyboard must be CLOAD"A", where "A" represents the letter identifying the program on the cassette tape.

Once a program has been loaded or inputted in the computer memory, it can be

listed on the CRT screen with the "LIST" command.

If only a portion of the program must be listed (usually because your program has more than 16 statements in it - the display capacity of the CRT screen), the format of the LIST command must be LISTA-B, where "A" equals the first statement number in a series to be listed and "B" equals the last statement number in a series to be listed.

If only one statement in your program is to be listed, the format of the LIST command must be LISTA, where "A" equals the statement number of the statement in your program to be listed.

The program currently in the computer memory can be executed with the "RUN" command.

If, during the execution of a program, you hit the BREAK key or the program executes a "STOP" instruction, you can restart the program execution at the next statement past the point you halted by keying in a "CONT" command.

Execution could be restarted at any point in the program by keying in the BASIC statement "GOTOA", where "A" represents the statement number in your program with which you wish to restart at.

Now, let's look at the various BASIC instructions that will fill the BASIC statements in your programs:

The "CLS" instruction will clear any data currently being displayed on the CRT screen.

The "GOTO" instruction causes the computer to resume execution at the statement number following the GOTO instruction (for example - 10 GOTO 100).

The "END" instruction stops the execution of your program but will not allow the use of a "CONT" command to restart the program.

The "STOP" instruction stops the execution of your program which can be restarted at the point it was stopped by the use of the "CONT" command.

The "REM" instruction is not really an instruction at all. It prefixes a section of text which is merely documentation for your program. Using REMs throughout a program, you can insert explanations concerning what your program is doing. For example - 10 REM THIS IS A COMMENT.

The "INPUT" instruction allows the user to receive the values to be stored in one or more fields used in your program. For example - 10 INPUT A,B,C will place a "?" prompt on the screen and then the computer will wait for the user to key in three numeric values separated by commas to be stored in fields A, B and C respectively. When the last of the three values is keyed in, hitting enter will store the values and resume program execution. Whatever you key in will appear on the screen after the "?" prompt.

The "INPUT#-1" instruction is similar to the regular INPUT instruction except that the field values are read from the cassette tape as opposed to the keyboard. When field values are inputted from a cassette tape, it is important that the

information was written onto the cassette tape in the same way it will be read back in later.

The instruction that writes field values to tape is "PRINT#-1". For example - if three numeric values were written to tape using:

```
10 PRINT#-1,A,B,C
```

the three numeric fields must be read back into a program with:

```
20 INPUT#-1,A,B,C
```

Another way to store values into fields used in your program is through the "READ", "DATA" and "RESTORE" instructions.

The READ instruction will receive field values supplied through DATA instructions embedded anywhere in your program. For example -

```
10 READ A,B,C
20 DATA 1,2,3
30 DATA 4,5,6
40 READ D,E,F
```

Here the READ of A,B and C will set them to 1, 2 and 3 respectively and the READ of D, E and F will set them to 4, 5 and 6 respectively.

Note that the DATA statements can be placed anywhere in the program without affecting program execution. The only rule reguarding this is that the DATA statements will be matched up with the READ statements encountered during execution in order by their statement numbers.

The only thing that can affect this rule is the "RESTORE" instruction. When this instruction is executed, the next DATA statement selected to be matched to an encountered READ instruction will be that DATA statement in your program with the lowest statement number. This allows you to reuse a series of DATA statements for tasks which must be executed several times.

If there are more READ instructions executed that there are DATA statements in your program without having executed a RESTORE instruction before this condition exists, an error will occur.

The "LET" instruction is used to assign a value to a field by arithmetic means. For example -

```
10 LET X=(((A+B)*C)-D)/E
```

This will add A and B together, multiply the result by C, subtract D from that result, divide that result by E and store the resultant in field X.

All the fields involved must be numeric fields that have been initialized with values of their own within your program. Note that E can not contain 0 because of being used as a divisor and note the use of parentheses to order the arithmetic operations to be performed during the value determination.

In the example - 20 LET A$ = "HELLO" , the word "HELLO" is stored in the string field A$.

The "LET" instruction also allows the use of some other functions. In LET X = ABS(A) , if A = -1, X would be set to 1 because of the absolute value function performed on A by ABS.

In LET X = INT(A) , if A = 1.5, X would be set to 1 because of the integer function performed on A by INT.

In LET X = RND(1) , X would be set to a random number between 0 and .999999 by the random number generator function RND(1).

In LET X = RND(A) , if A = 5, X would be set to a random number between 1 and 5 because of the random number function performed on A by RND.

In LET X = MEM , X would be set to the value corresponding to the number of bytes of free memory not being used for program statements and field storage. This can be useful in determining the maximum number of numeric fields that can be added to a table being used in your program. This can be determined by knowing that each additional numeric field in your table uses up 4 bytes of free memory.

The "IF" instruction allows you to compare two numeric fields or two string fields to each other for an equal to, greater than or a less than condition which will prompt a specific action.

In 10 IF A = B THEN 50 , A is compared to B for an equal condition. The "THEN" is a required keyword in the IF instruction which specifies the statement number at which the program will resume execution if the condition is true. The THEN action is the same as a GOTO instruction in this respect.

The "ON" instruction allows the program to select a statement number to GOTO or GOSUB depending on the value in a numeric field. In ON A GOTO 100, 200, 300 , if A contains 2 the instruction will GOTO 200. The GOTO in this instruction may be replaced by a GOSUB.

The "GOSUB" instruction is like a GOTO instruction in that the program will resume execution at the statement number specified in the instruction (for example - 10 GOSUB 200).

In this case however, the statement at which execution continues must be the first statement in a series of statements that performs a specific task within the program which might need to be performed at several places in the program.

This technique allows you to code the needed BASIC statements to perform that task at only one spot in your program and execute it by the use of GOSUB instructions at the numerous places in your program where it is needed.

This unique series of re-executed instructions are called a subroutine correctly and must end with a "RETURN" instruction. The RETURN instruction resumes execution of your program at the statement immediatly following the last GOSUB executed. The last GOSUB statement must be returned to because a subroutine might contain GOSUBs in order to perform its task.

The "FOR" instruction initiates a loop in your program which will re-execute a series of statements a specific number of times depending on a numeric count. In

```
10 FOR X = 0 TO A STEP B
20 PRINT X
30 NEXT X
```

if A contains 10 and B contains 2, the loop formed by statement numbers 10 thru 30 will execute statement 20 5 times. The loop control field would be initialized to 0 and B as specified by the STEP keyword would be added to X every time the NEXT instruction in statement 30 would be encountered during program execution.

A "NEXT" instruction must be the last instruction in a "FOR" loop series of instructions.

After the STEP value is added to the loop control field, the control field X would be compared to the terminating field "A". If a not equal condition exists, program execution resumes at the statement immediately following the last FOR statement. The last FOR statement must be gone to because a loop may contain a loop within its series of repetitive instructions.

The "PRINT" instruction displays a series of field values and/or literals accompanying the instruction on the next available print line on the CRT screen. In 10 PRINT "RATE"; R if R contains a 5.5, this will display "RATE 5.5" on the CRT screen.

A PRINT instruction may also contain a special function for inserting blanks between fields. In 20 PRINT "RATE"; TAB(A); R if A contains 10, the PRINT instruction will display 10 blanks between "RATE" and "5.5".

The "PRINTAT" instruction displays a series of fields and/or literals at a specific location on the current CRT screen image. In 30 PRINTAT A,"RATE";R if A contains a 64, the instruction will display "RATE 5.5" starting at the 64 byte in the screen area.

The screen area is mapped into 16 rows of 64 positions each, addressed for 0 in the upper left-hand corner to 1023 in the lower right-hand corner of the screen.

Lastly, there are several graphics instructions available. The CRT screen can be map ped graphically into an area of graphic points, 128 points horizontal addressed 0 to 127 left to right and 48 points vertical addressed 0 to 47 top to bottom.

To turn a point on, use the "SET" instruction. For example - 10 SET A,B where A contains the horizontal coordinate and B contains the vertical coordinate.

To turn a point off, use the "RESET" instruction. For example - 20 RESET A,B .

To test to see if a point is turned on, a special condition can be tested for. In 10 IF POINT(A,B) THEN 200 , if the point at the address specified by (A,B) is on then program execution would resume with statement number 200.

Combining these graphic instructions with PRINTAT instructions, impressive, self-explanatory displays can be generated.

In future articles, we will look at the capabilities of Level 2 BASIC and some comprehensive programming techniques reviews. The reason techniques are not stressed in Level 1 BASIC is due to its limited capabilities and infrequent use as a sole means of programming. Most people currently use Level 2 but a review of Level 1 BASIC is a first step toward a complete understanding of the TRS-80 from a programmer's standpoint.

.....

WORD SEARCH PUZZLE

THE WORDS TO FIND—

ANDROMEDA
AQUARIUS
ARIES
BOOTES
CANCER
CAPRICORNUS
CASSIOPEIA
CENTAURUS
CYGNUS
DRACO
GEMINI
LEO
LIBRA
ORION
PEGASUS
PISCES
SAGITTARIUS
SCORPIUS
TAURUS
VIRGO

YOUR PUZZLE—

EDHYOKBURCAPRICORNUS
ZWRUECBLPMTOCAQQKWSO
QSHAGNSUIRATTIGASNEL
ABTTCECHEENEDHOGRIVT
GFSFFOQCRKGCSFUUGWNF
YQIDMBPEGASUSVHORION
SBOOTESNQPSXRAKGYBDF
SLJPWSBTHUHDRDWEISIU
UUJYDQCAVHVAQEEMBNRO
QEBKRSHOISVCRMOIREIC
PISCESYRREOMFIHNNONL
YDNUHHEUDPEDAREIZNXS
AIEPOISSACIFBDNSDFRS
TYCDBZZPTPMUINCFWUUM
LLVJPCANCERMSAFYBIRW
QIRNXJEFNGKGSWNZGJLF
BZBMODCBILNHQTJAGNJA
DPLRYULBSURUATUGTGUI
ZAIRATDVPTCXZQJBJAJS
GRRSPOSQDNJAAAVAHVIB

SPECIAL!

********** TRS-80 USERS - TREAT YOUR COMPUTER TO THESE EXCITING PROGRAMS! **********

ENHANCED GRAPHICS AND MORE OPTIONS IN OUR CURRENT VERSIONS!

PRICED AS LOW AS 31¢ PER PROGRAM!

AIOS/1 - This program combines NATURAL LANGUAGE with action verb programming. 4K & up

NLOS/1 - Give your TRS-80 the power to read and understand ENGLISH! Build conversational data bases - solve problems and answer questions relating to information learned. 16K (NLOS/2 - 32K, enhanced, performs learned tasks, built-in vocabulary!)

MAZE/1 - Randomly generate and solve MAZES of selected complexity. 4K

CONSTELLATION - Unique graphics display the night sky, then travel to any star and view the night sky of that ALIEN planet. 16K (version 2, enhanced, built-in star chart!)

YG/1 - Players may challenge the TRS-80 to a game of YAHTZEE. 16K

CARTOON - Create and run ANIMATED PICTURES on your screen. 4K

CP/1 - Randomly generate and solve CROSSWORD PUZZLES - graphics. 16K

BGSG/1 - Command Colonial or Cylon fleets in BATTLESTAR GALACTICA. 16K

CHECKERS - Challenge your TRS-80 to a game of CHECKERS - graphics. 16K

LND/1 - Buy and manage properties as you build your REAL ESTATE empire. But beware, your tenants may give you trouble! 4K

MNP/1 - Challenge your TRS-80 to a game of MONOPOLY. 16K

SWG/1 - Challenge your TRS-80 to a CROSSWORD-like game. 4K

TRIVIA - Test your memory with this BRAIN TEASING game. 16K

POKER - Challenge your TRS-80 to a game of POKER. 4K

BWL/1 - Challenge your TRS-80 to a BOWLING match. 4K

CAL/1 - Turn your TRS-80 into a powerful CALCULATOR. 4K

TMT/1 - Chase a madman forward and backward in TIME. 4K

CLUE - Become a master DETECTIVE and solve murder mysteries. 4K

NB/1 - Hunt down the enemy's fleet in this exciting NAVAL BATTLE. 4K

AR/1 - Test your skill in a wrecked car in DEMOLITION DERBY. 4K

BB/1 - Challenge your TRS-80 to a BASEBALL game. 4K

CS/1 - Manage a nuclear power plant in CHINA SYNDROME. 4K

LL/1 - Attempt to land an extraterrestrial SPACE CRAFT. 4K

ENV/1 - Test your knowledge of ECOLOGY. 4K

RBT/1 - Guide a series of bombs in an attempt to blow up a group of invading ANDROIDS. 4K

WT/1 - Lead a WAGON TRAIN safely across the prairie. 4K

WAR/1 - Command airplanes, TANKS and ARMIES in a war. 4K

MS/1 - Test your MATH SKILLS at various levels of complexity. 4K

SHT/1 - Turn your TRS-80 into a SHOOTING GALLERY. 4K

PB/1 - Turn your TRS-80 into a PINBALL MACHINE. 4K

SW/1 - Engage in INTERSTELLAR CONFLICT against the Zetars. 16K

BNG/1 - Play an exciting game of BINGO with your TRS-80. 4K

GR/1 - Challenge your TRS-80 to a game of GIN RUMMY. 4K

BJ/1 - Challenge your TRS-80 to a game of BLACKJACK. 4K

PP/1 - Challenge your TRS-80 to a PING-PONG game. 4K

FOUR IN A ROW - Play the game of CONNECT FOUR - animated graphics. 16K

TWP/1 - Turn your TRS-80 into a powerful WORD PROCESSOR. 16K

TRG/1 - GENERATE REPORTS from tape data files with headings, control breaks, record selection and totals. 16K

TRS/1 - Multi-key field SORT UTILITY for tape data files. 16K

TDB/1 - Create, maintain and inquiry to TAPE DATA BASE files. 16K

INVASION - Prevent an ALIEN INVADER from destroying the Earth. 16K

LEVEL 2 BASIC! COMPLETE INSTRUCTIONS! CLOAD TESTED CASSETTES: $3.49! PROGRAM LISTINGS: $1.25! ORDER 2 OR MORE PROGRAMS - GET 10% OFF! ALL 41 PROGRAM LISTINGS FOR $12.95! PROGRAM LISTINGS EASILY ADAPTED TO PET, APPLE AND OTHER BASIC COMPUTERS!

SEND CHECK OR MONEY ORDER TO:

********** CYBERMATE * R.D.#3 BOX 192A * NAZARETH, PA 18064 **********

THE 80 NOTEBOOK
R.D.#3
Nazareth, PA 18064

BULK RATE
U.S. POSTAGE
PAID
Stockertown, PA 18083
Permit No. 8

4/80

3/81